Orienting (Hyper)graphs Under Explorable Stochastic Uncertainty

Evripidis Bampis
Sorbonne Université, CNRS, LIP6, France

Christoph Dürr
Sorbonne Université, CNRS, LIP6, France

Thomas Erlebach
School of Informatics, University of Leicester, United Kingdom

Murilo Santos de Lima
School of Informatics, University of Leicester, United Kingdom

Nicole Megow
Faculty of Mathematics and Computer Science, University of Bremen, Germany

Jens Schlöter
Faculty of Mathematics and Computer Science, University of Bremen, Germany

Abstract

Given a hypergraph with uncertain node weights following known probability distributions, we study the problem of querying as few nodes as possible until the identity of a node with minimum weight can be determined for each hyperedge. Querying a node has a cost and reveals the precise weight of the node, drawn from the given probability distribution. Using competitive analysis, we compare the expected query cost of an algorithm with the expected cost of an optimal query set for the given instance. For the general case, we give a polynomial-time $f(\alpha)$-competitive algorithm, where $f(\alpha) \in [1.618 + \epsilon, 2]$ depends on the approximation ratio α for an underlying vertex cover problem. We also show that no algorithm using a similar approach can be better than 1.5-competitive. Furthermore, we give polynomial-time 4/3-competitive algorithms for bipartite graphs with arbitrary query costs and for hypergraphs with a single hyperedge and uniform query costs, with matching lower bounds.

2012 ACM Subject Classification Theory of Computation → Design and analysis of algorithms; Mathematics of computing → Discrete mathematics

Keywords and phrases Explorable uncertainty, queries, stochastic optimization, graph orientation, selection problems

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.37

Related Version *Full Version*: https://arxiv.org/abs/2107.00572

Funding *Evripidis Bampis*: Supported by the grant ANR-19-CE48-0016 from the French National Research Agency (ANR)
Christoph Dürr: Supported by the grant ANR-19-CE48-0016 from the French National Research Agency (ANR)
Thomas Erlebach: Supported by EPSRC grant EP/S033483/1.
Murilo Santos de Lima: Funded by EPSRC grant EP/S033483/1.
Nicole Megow: Supported by the German Science Foundation (DFG) under contract ME 3825/1.
Jens Schlöter: Funded by the German Science Foundation (DFG) under contract ME 3825/1.

Acknowledgements The authors would like to thank the anonymous referees for their careful reading and helpful suggestions.

1 Introduction

The research area of explorable uncertainty is concerned with scenarios where parts of the input are initially uncertain, but the precise weight (or value) of an input item can be obtained via a *query*. For example, an uncertain weight may be represented as an interval that is guaranteed to contain the precise weight, but only a query can reveal the precise weight. Adaptive algorithms make queries one by one until they have gathered sufficient information to solve the given problem. The goal is to make as few queries as possible. In most of the previous work on explorable uncertainty, an adversarial model has been studied where an adversary determines the precise weights of the uncertain elements in such a way that the performance of the algorithm, compared to the optimal query set, is as bad as possible. While this model provides worst-case guarantees that hold for every possible instance, it is also very pessimistic because the adversary is free to choose the precise weights arbitrarily. In realistic scenarios, one may often have some information about where in the given interval the precise weight of an uncertain element is likely to lie. This information can be represented as a probability distribution and exploited in order to achieve better performance guarantees.

In this paper, we study the following problem under stochastic uncertainty: Given a family of (not necessarily disjoint) subsets of a set of uncertain elements, determine the element with minimum precise weight in each set, using queries of minimum total cost. Note that we do not necessarily need to obtain the precise minimum weight. We phrase the problem in the language of hypergraphs, where each uncertain element corresponds to a node and each set corresponds to a hyperedge. We call this the *hypergraph orientation* problem, as we can think of orienting each hyperedge towards its minimum-weight vertex. Each node $v \in V$ of a hypergraph $H = (V, E)$ is associated with a known continuous probability distribution[1] d_v over an interval $I_v = (\ell_v, r_v)$ and has query cost c_v. The precise weight of v is drawn independently from d_v and denoted by w_v. We assume that I_v is the minimal interval that contains the support of d_v, i.e., ℓ_v is the largest value satisfying $\mathbb{P}[w_v \leq \ell_v] = 0$ and r_v is the smallest value satisfying $\mathbb{P}[w_v \geq r_v] = 0$. For $S \subseteq V$, we define $c(S) = \sum_{v \in S} c_v$. An algorithm can sequentially make queries to vertices to learn their weights, until it has enough information to identify the minimum-weight vertex of each hyperedge. A query of v reveals its precise weight w_v, which is drawn independently from d_v. If all vertices have the same query cost, we say that the query costs are uniform and assume w.l.o.g. that $c_v = 1$ for all $v \in V$. Otherwise, we speak of arbitrary query costs. The objective of an algorithm is to minimize the expected cost of the queries it makes.

We also consider the special case where we are given a graph $G = (V, E)$ instead of a hypergraph $H = (V, E)$, called the *graph orientation* problem.

As an example consider a multi-national medical company that needs a certain product (say, chemical ingredient, medicine or vaccine) for its operation in each country. The particular products that are available in each country are different due to different approval mechanisms. The task is to find the best product for each country, that is, the best among the approved ones. The quality itself is independent of the country and can be determined by extensive tests in a lab (queries). The set of products available in one country corresponds to a hyperedge, and the problem of identifying the best product in each country is the hypergraph

[1] We assume the distribution is given in such a way that $\mathbb{P}[w_v \in (a, b)]$ can be computed in polynomial time for every $v \in V, a, b \in \mathbb{R}$. For all our algorithms it suffices to be given a probability matrix: rows correspond to vertices v, columns to elementary intervals (t_i, t_{i+1}), and entries to $\mathbb{P}[w_v \in (t_i, t_{i+1})]$, where $t_1, \ldots, t_{2|V|}$ represent the sorted elements of $\{\ell_v, r_v | v \in V\}$.

orientation problem.

Our contribution. Our main result (Section 3) is an algorithm for the graph orientation problem with competitive ratio $\frac{1}{2}(\alpha+\sqrt{8-\alpha(4-\alpha)})$, assuming we have an α-approximation for the vertex cover problem (which we need to solve on an induced subgraph of the given graph). This factor is always between $\phi \approx 1.618$ (for $\alpha = 1$), and 2 (for $\alpha = 2$). We show that, for the special cases of directing $\mathcal{O}(\log\ |V|)$ hyperedges and sorting $\mathcal{O}(1)$ sets, the algorithm can be applied with $\alpha = 1$ in polynomial running time. The algorithm has a preprocessing phase in two steps. First, we compute the probability that a vertex is *mandatory*, i.e., that it is part of any feasible solution, and we query all vertices with probability over a certain threshold. The second step uses a LP relaxation of the vertex cover problem to select some further vertices to query. Next, we compute an α-approximation of the vertex cover on a subgraph induced by the preprocessing, and we query the vertices in the given solution. The algorithm finishes with a postprocessing that only queries mandatory intervals. For the analysis, we show two main facts: (1) the expected optimal solution can be bounded by the expected optimal solutions for the subproblems induced by a partition of the vertices; (2) for the subproblem on which we compute a vertex cover, the expected optimal solution can be bounded by applying the Kőnig-Egerváry theorem [52] on a particular bipartite graph, in case of uniform costs. When given arbitrary query costs, we show in the full version [6] that we can utilize a technique of *splitting* the vertices in order to obtain a collection of disjoint stars with obvious vertex covers that imply a bound on the expected optimum.

We further show how to generalize the algorithm to hypergraphs. Unfortunately in this case it is #P-hard to compute the probability of a vertex being mandatory, but we can approximate it by sampling. This yields a randomized algorithm that attains, with high probability, a competitive ratio arbitrarily close to the expression given above for graphs. Here, we need to solve the vertex cover problem on an induced subgraph of an auxiliary graph that contains, for each hyperedge of the given hypergraph, all edges between the node with the leftmost interval and the nodes whose intervals intersect that interval.

We also consider a natural alternative algorithm (Section 4) that starts with a particular vertex cover solution followed by adaptively querying remaining vertices. We prove a competitive ratio of 4/3 on special cases, namely, for bipartite graphs with arbitrary cost and for a single hyperedge with uniform costs, and complement this by matching lower bounds.

Related work. Graph orientation problems are fundamental in the area of graph theory and combinatorial optimization. In general, graph orientation refers to the task of giving an orientation to edges in an undirected graph such that some given requirement is met. Different types of requirements have been investigated. While Robbins [50] initiated research on connectivity and reachability requirements already in the 1930s, most work is concerned with degree-constraints; cf. overviews given by Schrijver [52, Chap. 61] and Frank [29, Chap. 9].

Our requirement, orienting each edge towards its node with minimum weight, becomes challenging when there is uncertainty in the node weights. While there are different ways of modeling uncertainty in the input data, the model of explorable uncertainty was introduced by Kahan [40]. He considers the task of identifying the minimum element in a set of uncertainty intervals, which is equivalent to orienting a single hyperedge. Unlike in our model, no distributional information is known, and an adversary can choose weights in a worst-case manner from the intervals. Kahan [40] shows that querying the intervals in order of non-decreasing left endpoints requires at most one more query than the optimal query set, thus giving a competitive ratio of 2. Further, he shows that this is best possible in the

adversarial model.

Subsequent work addresses finding the k-th smallest value in a set of uncertainty intervals [27,37], caching problems [49], computing a function value [41], and classical combinatorial optimization problems, such as shortest path [26], knapsack [31], scheduling problems [2,3,21], minimum spanning tree and matroids [22,25,28,45,46]. Recent work on sorting elements of a single or multiple non-disjoint sets is particularly relevant as it is a special case of the graph orientation problem [24,38]. For sorting a single set in the adversarial explorable uncertainty model, there is a 2-competitive algorithm and it is best possible, even for arbitrary query costs [38]. The competitive ratio can be improved to 1.5 for uniform query cost by using randomization [38]. Algorithms with limited adaptivity have been proposed in [23].

Although the adversarial model is arguably pessimistic and real-world applications often come with some distributional information, surprisingly little is known on stochastic variants of explorable uncertainty. The only previous work we are aware of is by Chaplick et al. [16], in which they studied stochastic uncertainty for the problem of sorting a given set of uncertain elements, and for the problem of determining the minimum element in a given set of uncertain elements. They showed that the optimal decision tree (i.e., an algorithm that minimizes the expected query cost among all algorithms) for a given instance of the sorting problem can be computed in polynomial time. For the minimum problem, they leave open whether an optimal decision tree can be determined in polynomial time, but give a 1.5-competitive algorithm and an algorithm that guarantees a bound slightly smaller than 1.5 on the expectation of the ratio between the query cost of the algorithm and the optimal query cost. The problem of scheduling with testing [42] is also in the spirit of stochastic explorable uncertainty but less relevant here.

There are many other stochastic problems that take exploration cost into account. Some of the earliest work has studied multi-armed bandits [15,30,54] and Weitzman's Pandora's box problem [55], which are prime examples for analyzing the tradeoff between the cost for exploration and the benefit from exploiting gained information. More recently, query-variants of combinatorial optimization problems received some attention, in general [33,53], and for specific problems such as stochastic knapsack [20,43], orienteering [8,34], matching [5,7, 12,13,17], and probing problems [1,35,36]. Typically such work employs a *query-commit* model, meaning that queried elements must be part of the solution, or solution elements are required to be queried. These are quite strong requirements that lead to a different flavor of the cost-benefit tradeoff.

Research involving queries to identify particular graph structures or elements, or queries to verify certain properties, can be found in various flavors. A well-studied problem class is *property testing* [32], and there are many more, see e.g., [4,11,18,44,48,51]. Without describing such problems in detail, we emphasize a fundamental difference to our work. Typically, in these query models, the bounds on the number of queries made by an algorithm are *absolute numbers*, i.e., given as a function of the input size, but independent of the input graph itself and without any comparison to the minimum number of queries needed for the given graph.

2 Definitions and Preliminary Results

The hypergraph orientation problem and the graph orientation problem have already been defined in Section 1. In this section we first give additional definitions and discuss how we measure the performance of an algorithm. Then we introduce the concept of mandatory vertices and show how the probability for a vertex to be mandatory can be computed or

at least approximated efficiently. We also give a lower bound showing that no algorithm can achieve competitive ratio better than $\frac{4}{3}$. Next, based on the concept of witness sets, we define the vertex cover instance associated with an instance of our problem and define a class of vertex cover-based algorithms, which includes all the algorithms we propose in this paper. Finally, we characterize the optimal query set for each realization and give lower bounds on the expected optimal query cost, which we will use later in the analysis of our algorithms.

Definitions. To measure the performance of an algorithm, we compare the expected cost of the queries it makes to the expected optimal query cost. Formally, given a realization of the values, we call *feasible query set* a set of vertices to be queried that permits one to identify the minimum-weight vertex in every hyperedge. Note that a query set is feasible if, for each hyperedge, it either queries the node v with minimum weight w_v and all other nodes whose intervals contain w_v, or it does not query the node v with minimum weight but queries all nodes whose intervals overlap I_v, and in addition the precise weights of all those intervals lie to the right of I_v. An *optimal query set* is a feasible query set of minimum query cost. We denote by $\mathbb{E}[\mathrm{OPT}]$ the expected query cost of an optimal query set. Similarly, we denote by $\mathbb{E}[A]$ the expected query cost of the query set queried by an algorithm A. The supremum of $\mathbb{E}[A]/\mathbb{E}[\mathrm{OPT}]$, over all instances of the problem, is called the *competitive ratio* of A. Alternatively, one could compare $\mathbb{E}[A]$ against the cost $\mathbb{E}[A^*]$ of an optimal adaptive algorithm A^*. However, in explorable uncertainty, it is standard to compare against the optimal query set, and, since $\mathbb{E}[OPT]$ is a lower bound on $\mathbb{E}[A^*]$, all our algorithmic results translate to this alternative setting.

Let $F \in E$ be a hyperedge consisting of vertices $v_1, \ldots, v_k$, indexed in order of non-decreasing left endpoints of the intervals, i.e., $\ell_{v_1} \leq \ldots \leq \ell_{v_k}$. We call v_1 the *leftmost* vertex of F. We can assume that $I_{v_1} \cap I_{v_i} \neq \emptyset$ for all $2 \leq i \leq k$, because otherwise the vertex v_i could be removed from the hyperedge F. For the special case of graphs, this means that we assume $I_v \cap I_u \neq \emptyset$ for each $\{u, v\} \in E$, since otherwise we could simply remove the edge.

Mandatory vertices, probability to be mandatory. A vertex v is called *mandatory* if it belongs to every feasible query set for the given realization. For example, if for some edge $\{u, v\}$, vertex u has already been queried and its value w_u belongs to the interval I_v, then v is known to be mandatory. The following lemma was shown in [24] and fully characterizes mandatory vertices.

▶ **Lemma 2.1.** *A vertex $v \in V$ is mandatory if and only if there is a hyperedge $F \in E$ with $v \in F$ such that either (i) v is a minimum-weight vertex of F and $w_u \in I_v$ for some $u \in F \setminus \{v\}$, or (ii) v is not a minimum-weight vertex of F and $w_u \in I_v$ for the minimum-weight vertex u of F.*

For a hyperedge $F = \{v_1, \ldots, v_k\}$, where the vertices are again indexed by non-decreasing left endpoints, it was shown in [16, Section 3] that, if $I_{v_i} \subseteq I_{v_1}$ for some $2 \leq i \leq k$, then v_1 is mandatory for every realization. Thus, every algorithm can iteratively query all such elements in a preprocessing step, without worsening the competitive ratio. In the remainder of the paper, we assume w.l.o.g. that the instance under consideration is already preprocessed.

Similarly, if a hyperedge contains vertices u, v such that v has not been queried yet and is the leftmost vertex, while a query of u has revealed that $w_u \in I_v$, then it follows from Lemma 2.1 that v is mandatory for every realization of the unqueried vertices. The final stage of our algorithms will consist of querying mandatory vertices that are identified by this criterion, until the instance is solved.

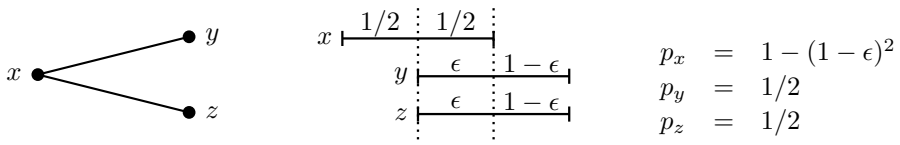

Figure 1 Instance and mandatory probabilities used in the proof of Theorem 2.3.

We denote by p_v the probability that a vertex v is mandatory. Querying vertices $v \in V$ that have a high probability p_v is a key element of our algorithms. For graphs, p_v is easy to compute as, by Lemma 2.1, v is mandatory iff $w_u \in I_v$ for some neighbor vertex u. Hence, $p_v = 1 - \prod_{u:\{u,v\}\in E} \mathbb{P}[w_u \notin I_v]$. For hypergraphs, however, we can show that the computation of p_v is #P-hard, even if all hyperedges have size 3. Luckily it is not difficult to get a good estimate of the probabilities to be mandatory for hypergraphs using sampling.

▶ **Lemma 2.2.** *There is a polynomial-time randomized algorithm that, given a hypergraph $H = (V, E)$, a vertex $v \in V$, and parameters $\epsilon, \delta \in (0, 1)$, produces a value y such that $|y - p_v| \geq \epsilon$ with probability at most δ. Its time complexity is $O(|V| \ln(1/\delta)/\epsilon^2)$.*

General lower bound. We have the following lower bound.

▶ **Theorem 2.3.** *Every algorithm for the graph orientation problem has competitive ratio at least $\frac{4}{3}$, even for uniform query costs and even if no restriction on the running time of the algorithm is imposed.*

Proof. Consider three vertices x, y, z, with $I_x = (0, 2)$ and $I_y = I_z = (1, 3)$, and uniform query costs $c_x = c_y = c_z = 1$. The only edges are $\{x, y\}$ and $\{x, z\}$. The probabilities are such that $\mathbb{P}[w_x \in (1, 2)] = \frac{1}{2}$ and $\mathbb{P}[w_y \in (1, 2)] = \mathbb{P}[w_z \in (1, 2)] = \epsilon$, for some $0 < \epsilon \ll \frac{1}{2}$; see Figure 1. If $w_x \in (0, 1]$, which happens with probability $\frac{1}{2}$, querying x is enough. If $w_x \in (1, 2)$ and $w_y, w_z \in [2, 3)$, which happens with probability $\frac{1}{2}(1-\epsilon)^2$, querying y and z is enough. Otherwise, all three vertices must be queried. We have

$$\mathbb{E}[\mathrm{OPT}] = \frac{1}{2} \cdot 1 + \frac{1}{2}(1-\epsilon)^2 \cdot 2 + \frac{1}{2}\left(1 - (1-\epsilon)^2\right) \cdot 3 = 2 - \frac{(1-\epsilon)^2}{2},$$

which tends to $\frac{3}{2}$ as ϵ approaches 0. Since y and z are identical and we can assume that an algorithm always queries first a vertex that it knows to be mandatory (if there is one), we only have three possible decision trees to consider:

1. First query x; if $w_x \in (1, 2)$, then query y and z. The expected query cost is 2.
2. First query y. If $w_y \in (1, 2)$, then query x, and query z if $w_x \in (1, 2)$. If $w_y \in [2, 3)$, then query z, and query x if $w_z \in (1, 2)$. The expected query cost is $1 + \frac{3}{2}\epsilon + (1-\epsilon)(1+\epsilon)$, which tends to 2 as ϵ approaches 0.
3. First query y. Whatever happens, query x, then query z if $w_x \in (1, 2)$. The expected query cost is $\frac{5}{2}$, so this is never better than the previous options.

With either choice (even randomized), the competitive ratio tends to at least $\frac{4}{3}$ as $\epsilon \to 0$. ◀

This lower bound can be adapted for a single hyperedge $\{x, y, z\}$. For arbitrary query costs, it works even for a single edge $\{x, y\}$, by taking $c_x = 1$ and $c_y = 2$.

Witness sets, vertex cover instance, vertex cover-based algorithms. Another key concept of our algorithms is to exploit *witness sets* [14, 25]. A subset $W \subseteq V$ is a *witness set* if $W \cap Q \neq \emptyset$ for all feasible query sets Q. The following lemma was shown in [40].

▶ **Lemma 2.4.** *Let $F = \{v_1, \ldots, v_k\}$ be a hyperedge, and let v_1 be the leftmost vertex of F. Then $\{v_1, v_i\}$ is a witness set for each $2 \leq i \leq k$.*

The lemma implies that one can obtain a 2-competitive algorithm in the adversarial model for the hyperedge orientation problem: For uniform query costs, it suffices to repeatedly query witness sets of size 2 until the instance is solved, by a standard witness set argument [25]. For arbitrary query costs, this approach can be combined with the local ratio technique [9] to obtain the same competitive ratio (in a similar way as done in [38] for the sorting problem). Our goal is to achieve better competitive ratios in the stochastic setting. Motivated by Lemma 2.4, we can now define the *vertex cover instance*.

▶ **Definition 2.5.** *Given a hypergraph $H = (V, E)$, the* vertex cover instance *of H is the graph $\bar{G} = (V, \bar{E})$ with $\{v, u\} \in \bar{E}$ if and only if there is a hyperedge $F \in E$ such that $v, u \in F$, v is leftmost in F and $I_v \cap I_u \neq \emptyset$. For the special case of a graph G instead of a hypergraph H, it holds that $\bar{G} = G$.*

Since each edge of the vertex cover instance $\bar{G}$ is a witness set by Lemma 2.4, we can observe that each feasible query set Q is a vertex cover of $\bar{G}$. Using the vertex cover instance, we can define a class of algorithms for the hypergraph orientation problem as follows: An algorithm is *vertex cover-based* if it implements the following pattern:

1. Non-adaptively query a vertex cover VC of $\bar{G}$;
2. Iteratively query mandatory vertices until the minimum-weight vertex of each hyperedge is known: For each hyperedge $F \in E$ for which the minimum weight is still unknown, query the vertices in order of left endpoints until the minimum weight is found.

By definition of the second step, each vertex cover-based algorithm clearly orients each hyperedge. Furthermore, Lemma 2.1 implies that each vertex queried in the last step is indeed mandatory for all realizations that are consistent with the currently known information, i.e., the weights of the previously queried vertices. For graphs, this is easy to see, and for hypergraphs, this can be shown as follows: For a hyperedge F that isn't solved after the first step and has leftmost vertex v initially, the vertex cover VC has queried v or all other vertices of F. In the latter case, v is the only unqueried vertex of F and I_v must contain the precise weight of one of the other vertices, hence v is mandatory. In the former case, the remaining candidates for being the minimum-weight vertex are (1) the vertex with leftmost precise weight among those queried in the first step, and (2) the unqueried vertices whose intervals contain that precise weight. It is then clear that the leftmost vertex is mandatory, and querying it either solves the hyperedge or yields a situation of the same type.

All the algorithms we propose in this paper are vertex cover-based. We have the following lower bounds for vertex cover-based algorithms.

▶ **Theorem 2.6.** *No vertex cover-based algorithm has competitive ratio better than $\frac{3}{2}$ for the hypergraph orientation problem. This result holds even in the following special cases:*

1. *The graph has only a single hyperedge but the query costs are not uniform.*
2. *The query costs are uniform and the vertex cover instance $\bar{G}$ is bipartite.*
3. *The instance is a non-bipartite graph orientation instance with uniform query costs.*

We remark that the second step of vertex cover-based algorithms must be adaptive: In the full version we show that any algorithm consisting of two non-adaptive stages cannot have competitive ratio $o(\log n)$, even for a single hyperedge with n vertices and uniform query costs.

Bounds on $\mathbb{E}[\mathrm{OPT}]$. Let $\mathcal{R}$ be the set of all possible realizations and let $\mathrm{OPT}(R)$ for $R \in \mathcal{R}$ be the optimal query cost for realization R. As each feasible query set Q must include a vertex cover of $\bar{G}$, the minimum weight of a vertex cover of $\bar{G}$ (using the query costs as weights) is a lower bound on the optimal query cost for each realization and, thus, on $\mathbb{E}[\mathrm{OPT}]$. This observation in combination with Lemma 2.1 also gives us a way to identify an optimal query set for a fixed realization, by using the knowledge of the exact vertex weights.

▶ **Observation 2.7.** *For a fixed realization R of an instance of the hypergraph orientation problem, let M be the set of vertices that are mandatory (cf. Lemma 2.1), and let VC_M be a minimum-weight vertex cover of $\bar{G}[V \setminus M]$. Then $M \cup VC_M$ is an optimal query set for realization R.*

Computing $\mathrm{OPT}(R)$ for a fixed and known realization R is NP-hard [24]. This extends to the hypergraph orientation problem and the computation of $\mathbb{E}[\mathrm{OPT}]$: We can reduce from the problem of computing $\mathrm{OPT}(R)$ by concentrating the probability mass of all intervals onto the weights in realization R. The reduction of [24] in combination with [19] also implies APX-hardness.

To analyze the performance of our algorithms, we compare the expected cost of the algorithms to the expected cost of the optimal solution. By Observation 2.7, $c(M) + c(VC_M)$ is the minimum query cost for a fixed realization R, where $M \subseteq V$ is the set of mandatory elements in the realization and VC_M is a minimum-weight vertex cover for the subgraph $\bar{G}[V \setminus M]$ of the vertex cover instance $\bar{G} = (V, \bar{E})$ induced by $V \setminus M$. Thus, the optimal solution for a fixed realization is completely characterized by the set of mandatory elements in the realization. Using this, we can characterize $\mathbb{E}[\mathrm{OPT}]$ as $\mathbb{E}[\mathrm{OPT}] = \sum_{M \subseteq V} p(M) \cdot c(M) + \sum_{M \subseteq V} p(M) \cdot c(VC_M)$, where $p(M)$ denotes the probability that M is the set of mandatory elements. It follows that $\sum_{M \subseteq V} p(M) \cdot c(M) = \sum_{v \in V} p_v \cdot c(v)$, since both terms describe the expected cost for querying mandatory elements, which leads to the following characterization of $\mathbb{E}[\mathrm{OPT}]$:

$$\mathbb{E}[\mathrm{OPT}] = \sum_{v \in V} p_v \cdot c_v + \sum_{M \subseteq V} p(M) \cdot c(VC_M).$$

A key technique for our analysis is lower bounding $\mathbb{E}[\mathrm{OPT}]$ by partitioning the optimal solution into subproblems and discarding dependencies between elements in different subproblems.

▶ **Definition 2.8.** *For a realization R and any subset $S \subseteq V$, let $\mathrm{OPT}_S = \min_{Q \in \mathcal{Q}} c(Q \cap S)$, where $\mathcal{Q}$ is the set of all feasible query sets for realization R.*

▶ **Lemma 2.9.** *Let $S_1, \ldots, S_k$ be a partition of V. Then $\mathbb{E}[\mathrm{OPT}] \geq \sum_{i=1}^{k} \mathbb{E}[\mathrm{OPT}_{S_i}]$.*

Proof. We start the proof by characterizing $\mathbb{E}[\mathrm{OPT}_{S_i}]$ for each $i \in \{1, \ldots, k\}$. Let $R \in \mathcal{R}$ be a realization in which M is the set of mandatory elements. Then OPT_{S_i} needs to contain all mandatory elements of S_i, and resolve all remaining dependencies between vertices of S_i, i.e., query a minimum-weight vertex cover $VC_M^{S_i}$ for the subgraph $\bar{G}[S_i \setminus M]$. Thus, it follows

$$\mathbb{E}[\mathrm{OPT}_{S_i}] = \sum_{v \in S_i} p_v \cdot c_v + \sum_{M \subseteq V} p(M) \cdot c(VC_M^{S_i}). \tag{1}$$

By summing Equation (1) over all $i \in \{1, \dots, k\}$, we obtain the lemma:

$$\begin{aligned}\sum_{i=1}^{k} \mathbb{E}[\mathrm{OPT}_{S_i}] &= \sum_{i=1}^{k} \left(\sum_{v \in S_i} p_v \cdot c_v + \sum_{M \subseteq V} p(M) \cdot c(VC_M^{S_i}) \right) \\ &= \sum_{v \in V} p_v \cdot c_v + \sum_{M \subseteq V} p(M) \cdot \left(\sum_{i=1}^{k} c(VC_M^{S_i}) \right) \\ &\leq \sum_{v \in V} p_v \cdot c_v + \sum_{M \subseteq V} p(M) \cdot c(VC_M) = \mathbb{E}[\mathrm{OPT}],\end{aligned}$$

where the second equality follows from $S_1, \dots, S_k$ being a partition. The inequality follows from $\sum_{i=1}^{k} c(VC_M^{S_i})$ being the cost of a minimum weighted vertex cover for a subgraph of $G[V \setminus M]$, while $c(VC_M)$ is the minimum cost for a vertex cover of the whole graph. ◀

For the case of arbitrary query costs, we will sometimes need to partition V in such a way that a vertex v can be split into fractions that are in different parts of the partition. We view each fraction as a copy of v, and the split is done in such a way that the query costs of all copies of v add up to c_v. Further, the probability distribution for being mandatory in the resulting instance is such that either all copies of v are mandatory or none of them is, and the former happens with probability p_v. (A detailed discussion of this process can be found in the full version.) We refer to the application of this operation to a vertex as a *vertex split* and note that it can be applied repeatedly.

▶ **Observation 2.10.** *Let* OPT' *be the optimal solution for an instance that is created by iteratively executing vertex splits. Then,* $\mathbb{E}[\mathrm{OPT}'] = \mathbb{E}[\mathrm{OPT}]$. *Furthermore, Lemma 2.9 also applies to* $\mathbb{E}[\mathrm{OPT}']$ *and the modified instance.*

3 A Threshold Algorithm for Orienting Hypergraphs

We present an algorithm for orienting graphs and its generalization to hypergraphs.

3.1 Orienting Graphs

We consider the graph orientation problem. As a subproblem, we solve a vertex cover problem. This problem is NP-hard and 2-approximation algorithms are known [56]. For several special graph classes, there are improved algorithms [39]. Using an α-approximation as a black box, we give a competitive ratio between $\phi \approx 1.618$ ($\alpha = 1$) and 2 ($\alpha = 2$) as a function depending on α.

Algorithm 1 Threshold

Input: Instance $G = (V, E)$, p_v for each $v \in V$, parameter $d \in [0, 1]$, and an α-approximation black box for the vertex cover problem

1 Let $M = \{v \in V \mid p_v \geq d\}$;
2 Solve (LP) for $G[V \setminus M]$ and let x^* be an optimal basic feasible solution;
3 Let $V_1 = \{v \in V \mid x_v^* = 1\}$ and similarly $V_{1/2}, V_0$;
4 Use the α-approximation black box to approximate a vertex cover VC' for $G[V_{1/2}]$;
5 Query $Q = M \cup V_1 \cup VC'$; /* Q is a vertex cover of G */
6 Query the mandatory elements of $V \setminus Q$;

Algorithm 1 is parameterized by a threshold $d \in [0,1]$, which is optimized depending on the approximation ratio α of the chosen vertex cover procedure. The algorithm executes a preprocessing of the vertex cover instance by using the following classical LP relaxation, for which each optimal basic feasible solution is half-integral [47]:

$$\begin{array}{lll} \min & \sum_{v\in V} c_v \cdot x_v & \\ \text{s.t.} & x_v + x_u \geq 1 & \forall \{u,v\} \in E \\ & x_v \geq 0 & \forall v \in V \end{array} \tag{LP}$$

▶ **Theorem 3.1.** *Given an α-approximation with $1 \leq \alpha \leq 2$ for the vertex cover problem (on the induced subgraph $G[V_{1/2}]$, see Line 4),* Threshold *with parameter d achieves a competitive ratio of $\max\{\frac{1}{d}, \alpha + (2-\alpha)\cdot d\}$ for the graph orientation problem. Optimizing d yields a competitive ratio of $\frac{1}{2}(\alpha + \sqrt{8-\alpha(4-\alpha)})$.*

Proof. Here, we show the result for uniform query costs. The generalization to arbitrary query costs requires an additional technical step involving vertex splitting and is discussed in the full version. Since Q is a vertex cover for G, querying it in Line 5 and resolving all remaining dependencies in Line 6 clearly solves the graph orientation problem. Note that $V \setminus Q$ is an independent set in G, and thus the nodes in $V \setminus Q$ can only be made mandatory by the results of the queries to Q. Hence, it is known after Line 5 which nodes in $V \setminus Q$ are mandatory, and they can be queried in Line 6 in arbitrary order (or in parallel).

We continue by showing the competitive ratio of $\max\{\frac{1}{d}, \alpha + (2-\alpha)\cdot d\}$. Algebraic transformations show that the optimal choice for the threshold is $d(\alpha) = 2/(\alpha + \sqrt{8-\alpha(4-\alpha)})$. The desired competitive ratio for Threshold with $d = d(\alpha)$ follows.

The algorithm queries set Q and all other vertices only if they are mandatory, hence

$$\mathbb{E}[ALG] = |Q| + \sum_{v\in V\setminus Q} p_v = |M| + |V_1| + |VC'| + \sum_{v\in V_0} p_v + \sum_{v\in V_{1/2}\setminus VC'} p_v. \tag{2}$$

The expected optimal cost can be lower bounded by partitioning and Lemma 2.9:

$$\mathbb{E}[\mathrm{OPT}] \geq \mathbb{E}[\mathrm{OPT}_M] + \mathbb{E}[\mathrm{OPT}_{V_1\cup V_0}] + \mathbb{E}[\mathrm{OPT}_{V_{1/2}}]. \tag{3}$$

In the remainder we compare $\mathbb{E}[ALG]$ with $\mathbb{E}[\mathrm{OPT}]$ component-wise.

We can lower bound $\mathbb{E}[\mathrm{OPT}_M]$ by $\sum_{v\in M} p_v$ using Equation (1). By definition of M, it holds that $\mathbb{E}[\mathrm{OPT}_M] \geq \sum_{v\in M} p_v \geq d \cdot |M|$. Thus,

$$|M| \leq \frac{1}{d} \cdot \mathbb{E}[\mathrm{OPT}_M]. \tag{4}$$

Next, we compare $|V_1| + \sum_{v\in V_0} p_v$ with $\mathbb{E}[\mathrm{OPT}_{V_1\cup V_0}]$. For this purpose, let $G[V_1 \cup V_0]$ be the subgraph of G induced by $V_1 \cup V_0$, and let $G'[V_1 \cup V_0]$ be the bipartite graph that is created by removing all edges between elements of V_1 from $G[V_1 \cup V_0]$. It follows from similar arguments as in [47, Theorem 2] that V_1 is a minimum vertex cover of $G'[V_1 \cup V_0]$. This allows us to apply the famous Kőnig-Egerváry theorem [52]. By the latter there is a matching h mapping each $v \in V_1$ to a distinct $h(v) \in V_0$ with $\{v, h(v)\} \in E$. Denoting $S = \{h(v) \mid v \in V_1\}$, we can infer $\mathbb{E}[\mathrm{OPT}_{V_1\cup V_0}] \geq \mathbb{E}[\mathrm{OPT}_{V_1\cup S}] + \mathbb{E}[\mathrm{OPT}_{V_0\setminus S}]$.

Any feasible solution must query at least one endpoint of all edges of the form $\{v, h(v)\}$. This implies $\mathbb{E}[\mathrm{OPT}_{V_1\cup S}] \geq |V_1|$. Since additionally $p_{h(v)} \leq d$ for each $h(v) \in S$, we get

$$\sum_{v\in V_1} \left(1 + p_{h(v)}\right) \leq (1+d)\cdot |V_1| \leq (1+d)\cdot \mathbb{E}[OPT_{V_1\cup S}]. \tag{5}$$

By lower bounding $\mathbb{E}[OPT_{V_0 \setminus S}]$ with $\sum_{v \in V_0 \setminus S} p_v$ and using (5), we get

$$|V_1| + \sum_{v \in V_0} p_v = \sum_{v \in V_1} \left(1 + p_{h(v)}\right) + \sum_{v \in V_0 \setminus S} p_v \tag{6}$$
$$\leq (1+d) \cdot \mathbb{E}[\mathrm{OPT}_{V_1 \cup S}] + \mathbb{E}[\mathrm{OPT}_{V_0 \setminus S}] \leq (1+d) \cdot \mathbb{E}[\mathrm{OPT}_{V_1 \cup V_0}].$$

Finally, consider the term $|VC'| + \sum_{v \in V_{1/2} \setminus VC'} p_v$. Let VC^* be a minimum cardinality vertex cover for $G[V_{1/2}]$. Then, it holds $|VC^*| \geq \frac{1}{2} \cdot |V_{1/2}|$. This is, because in the optimal basic feasible solution to the LP relaxation x^*, each vertex in $V_{1/2}$ has a value of $\frac{1}{2}$. A vertex cover with $|VC^*| < \frac{1}{2} \cdot |V_{1/2}|$ would contradict the optimality of x^*. The following part of the analysis crucially relies on $|VC^*| \geq \frac{1}{2} \cdot |V_{1/2}|$, which is the reason why Threshold executes the LP relaxation-based preprocessing before applying the α-approximation.

Now, the expected cost of the algorithm for the subgraph $G[V_{1/2}]$ is $|VC'| + \sum_{v \in I'} p_v \leq |VC'| + d \cdot |I'|$ with $I' = V_{1/2} \setminus VC'$. Since $|VC'| \geq |VC^*| \geq \frac{1}{2} \cdot |V_{1/2}|$, there is a tradeoff between the quality of $|VC'|$ and the additional cost of $d \cdot |I'|$. If $|VC'|$ is close to $\frac{1}{2} \cdot |V_{1/2}|$, then it is close to $|VC^*|$ but, on the other hand, $|I'|$ then is close to $\frac{1}{2} \cdot |V_{1/2}|$, which means that the additional cost $d \cdot |I'|$ is high. Vice versa, if the cost for $|VC'|$ is high because it is larger than $\frac{1}{2} \cdot |V_{1/2}|$, then $|I'|$ is close to zero and the additional cost $d \cdot |I'|$ is low. We exploit this tradeoff and upper bound $\frac{|VC'| + d \cdot |I'|}{|VC^*|}$ in terms of the approximation factor α of the vertex cover approximation. Assume that the approximation factor α is tight, i.e., $|VC'| = \alpha \cdot |VC^*|$. Since $d \leq 1$, this is the worst case for the ratio $\frac{|VC'| + d \cdot |I'|}{|VC^*|}$. (In other words, if the approximation factor was not tight, we could replace α by the approximation factor that is actually achieved and carry out the following calculations with that smaller value of α instead, yielding an even better bound.) Using $|VC'| = \alpha \cdot |VC^*|$ and $|VC^*| \geq \frac{1}{2} \cdot |V_{1/2}|$, we can derive

$$|I'| = |V_{1/2}| - |VC'| = |V_{1/2}| - \alpha \cdot |VC^*| \leq (2 - \alpha) \cdot |VC^*|.$$

For the cost of the algorithm for subgraph $G[V_{1/2}]$ we get

$$|VC'| + d \cdot |I'| \leq \alpha \cdot |VC^*| + d \cdot (2 - \alpha) \cdot |VC^*| = (\alpha + (2 - \alpha) \cdot d) \cdot |VC^*|.$$

Since $|VC^*| \leq \mathbb{E}[\mathrm{OPT}_{V_{1/2}}]$, we get

$$|VC'| + d \cdot |I'| \leq (\alpha + (2 - \alpha) \cdot d) \cdot \mathbb{E}[\mathrm{OPT}_{V_{1/2}}]. \tag{7}$$

Combining Equations (2), (4), (6) and (7), we can upper bound the cost of the algorithm:

$$\mathbb{E}[ALG] = |M| + |V_1| + \sum_{v \in V_0} p_v + |VC'| + \sum_{v \in V_{1/2} \setminus VC'} p_v.$$
$$\leq \frac{1}{d} \cdot \mathbb{E}[\mathrm{OPT}_M] + (1+d) \cdot \mathbb{E}[\mathrm{OPT}_{V_1 \cup V_0}] + (\alpha + (2-\alpha) \cdot d) \cdot \mathbb{E}[\mathrm{OPT}_{V_{1/2}}]$$
$$\leq \max\left\{\frac{1}{d}, (1+d), (\alpha + (2-\alpha) \cdot d)\right\} \cdot \mathbb{E}[\mathrm{OPT}],$$

where the last inequality follows from the lower bound on OPT in (3). Observe that for any $d \in [0,1]$ and $\alpha \in [1,2]$, it holds that $(\alpha + (2-\alpha) \cdot d) \geq (1+d)$. We conclude with $\mathbb{E}[ALG] \leq \max\{\frac{1}{d}, (\alpha + (2-\alpha) \cdot d)\} \cdot \mathbb{E}[\mathrm{OPT}]$, which implies the theorem. ◀

In the full version of the paper, we show that the analysis of Threshold is tight. To benefit from a better approximation factor than $\alpha = 2$ for solving the minimum vertex cover

problem, we would need to know in advance the specialized graph class on which we want to solve this subproblem. In some cases, we can benefit from optimal or approximation algorithms, e.g., using Threshold with the PTAS for planar graphs [10] allows us to achieve a competitive ratio of at most $1.618 + \epsilon$, for any $\epsilon > 0$, if the input graph is planar.

3.2 Orienting Hypergraphs

▶ **Theorem 3.2.** *Given an α-approximation with $1 \leq \alpha \leq 2$ for the vertex cover problem (on the induced subgraph $\bar{G}[V_{1/2}]$ of the vertex cover instance given by Definition 2.5, cf. Line 4), a modified version of the* Threshold *algorithm solves the hypergraph orientation problem with arbitrary query costs with competitive ratio $R = \frac{1}{2}\left(\alpha + \sqrt{\alpha^2 + 4(2-\alpha)(1+\alpha\epsilon+(2-\alpha)\epsilon^2)} + (4-2\alpha)\epsilon\right)$ with probability at least $1-\delta$. Its running time is upper bounded by the complexity of the sampling procedure and the vertex cover black box procedure.*

Proof. The modified algorithm works with the vertex cover instance $\bar{G}$ instead of the given hypergraph H, following Definition 2.5. In Line 1, we use $d(\alpha) = 1/R + \epsilon$. In Line 6, we iteratively query mandatory vertices until the instance is solved. In addition, we use a random estimation Y_v of p_v, instead of the precise probability, using the procedure described in Lemma 2.2 with parameters ϵ and δ' such that $1-\delta = (1-\delta')^n$. As a result, with probability at least $1-\delta$ we have that for every vertex v, the estimation Y_v has absolute error at most ϵ. In case of this event we obtain the following bound on the cost (which is optimized for the chosen value of d), namely $\mathbb{E}[\text{ALG}] \leq \max\{\frac{1}{d-\epsilon}, (1+d+\epsilon), (\alpha+(2-\alpha)(d+\epsilon))\} \cdot \mathbb{E}[\text{OPT}]$. ◀

Sorting a set of elements is equivalent to determining, for each pair of elements, which of the two has smaller weight. Hence, the problem of sorting multiple sets of elements with uncertain weights is a special case of the graph orientation problem: For each set to be sorted, the edge set of a complete graph on its elements is added to a graph, and the resulting instance of the graph orientation problem is then equivalent to the given instance of the sorting problem. We can show the following theorem.

▶ **Theorem 3.3.** *For the special cases of orienting $\mathcal{O}(\log |V|)$ hyperedges and sorting $\mathcal{O}(1)$ sets,* Threshold *can be applied with $\alpha = 1$ in polynomial running time.*

4 Vertex Cover-Based Algorithms: Improved Results for Special Cases

Consider an arbitrary vertex cover-based algorithm ALG. It queries a vertex cover VC in the first stage, and continues with elements of $V \setminus VC$ if they are mandatory. Thus,

$$\begin{aligned}\mathbb{E}[\text{ALG}] &= c(VC) + \sum_{v \in V \setminus VC} p_v \cdot c_v = \sum_{v \in VC} \big(p_v \cdot c_v + (1-p_v)\cdot c_v\big) + \sum_{v \in V \setminus VC} p_v \cdot c_v \\ &= \sum_{v \in V} p_v \cdot c_v + \sum_{v \in VC} (1-p_v)\cdot c_v.\end{aligned}$$

Since the first term is independent of VC, ALG is the best possible vertex cover-based algorithm if it minimizes $\sum_{v \in VC}(1-p_v)\cdot c_v$. We refer to this algorithm as BestVC.

To implement BestVC, we need the exact value p_v, for all $v \in V$, and an optimal algorithm for computing a weighted vertex cover. As mentioned in Section 2, the first problem is #P-hard in hypergraphs, but it can be solved exactly in polynomial time for graphs. The weighted vertex cover problem can be solved optimally in polynomial time for bipartite graphs.

In general, BestVC has competitive ratio at least 1.5 (Theorem 2.6). However, we show in the following that it is 4/3-competitive for two special cases. It remains open whether BestVC still outperforms Threshold if the vertex cover is only approximated with a factor $\alpha > 1$.

4.1 A Best Possible Algorithm for Orienting Bipartite Graphs

▶ **Theorem 4.1.** BestVC *is* $\frac{4}{3}$*-competitive for the bipartite graph orientation problem.*

Proof. In the full version of the paper, we show that BestVC is $\frac{4}{3}$-competitive for the problem of orienting stars if both vertex cover-based algorithms (either querying the leaves or the center first) have the same expected cost. In this proof, we divide the instance into subproblems that fulfill these requirements, and use the result for stars to infer $\frac{4}{3}$-competitiveness for bipartite graphs.

Let VC be a minimum-weight vertex cover (with weights $c_v \cdot (1 - p_v)$) as computed by BestVC in the first phase. By the Kőnig-Egerváry theorem (e.g., [52]), there is a function $\pi : E \to \mathbb{R}$ with $\sum_{\{u,v\}\in E} \pi(u,v) \leq c_v \cdot (1 - p_v)$ for each $v \in V$. By duality theory, the constraint is tight for each $v \in VC$, and $\pi(u,v) = 0$ holds if both u and v are in VC. Thus, we can interpret π as a function that distributes the weight of each $v \in VC$ to its neighbors outside of VC.

For each $v \in VC$ and $u \in V \setminus VC$, let $\lambda_{u,v} := \frac{\pi(u,v)}{(1-p_u)\cdot c_u}$ denote the fraction of the weight of u that is used by π to cover the weight of v. Moreover, for $u \in V \setminus VC$, let $\tau_u := 1 - \sum_{\{u,v\}\in E} \lambda_{u,v}$ be the fraction of the weight of u that is not used by π to cover the weight of any $v \in VC$. Then, we can write the expected cost of BestVC as follows:

$$\mathbb{E}[\text{BestVC}] = \sum_{v\in VC} \left(c_v + \sum_{u\in V\setminus VC} p_u \cdot \lambda_{u,v} \cdot c_u \right) + \sum_{u\in V\setminus VC} p_u \cdot \tau_u \cdot c_u. \tag{8}$$

Using Observation 2.10, we compare $\mathbb{E}[\text{BestVC}]$ with the expected optimum $\mathbb{E}[\text{OPT}']$ for an instance $G' = (V', E', c')$ that is created by splitting vertices. We modify the mandatory distribution as in Section 2, which implies $\mathbb{E}[\text{OPT}] = \mathbb{E}[\text{OPT}']$. We add the following copies:

1. For each $v \in VC$, we add a copy v' of v to V' with $c'_{v'} = c_v$.
2. For all $u \in V \setminus VC$ and $v \in VC$ with $\lambda_{u,v} > 0$, we add a copy u'_v of u to V' with $c'_{u'_v} = \lambda_{u,v} \cdot c_u$.
3. For each $u \in V \setminus VC$ with $\tau_u > 0$, we add a copy u' of u to V' with $c'_{u'} = \tau_u \cdot c_u$.

Let p'_v denote the probability of v being mandatory for instance G'. By definition of the vertex split operation, we have $p_v = p'_u$ for each $v \in V$ and each copy $u \in V'$ of v. Further, for each $v \in VC$, define $H'_v = \{u'_v \mid u \in V \text{ with } \lambda_{u,v} > 0\}$. By definition of H'_v and G', all H'_v are pairwise disjoint. Let $\mathcal{H}' = \bigcup_{v\in VC} \left(H'_v \cup \{v'\}\right)$; then we can express $\mathbb{E}[\text{BestVC}]$ as follows:

$$\begin{aligned}\mathbb{E}[\text{BestVC}] &= \sum_{v\in VC} \left(c_v + \sum_{u\in V\setminus VC} p_u \cdot \lambda_{u,v} \cdot c_u \right) + \sum_{u\in V\setminus VC} p_u \cdot \tau_u \cdot c_u \\ &= \sum_{v\in VC} \left(c'_{v'} + \sum_{u\in H'_v} p'_u \cdot c'_u \right) + \sum_{u\in V'\setminus\mathcal{H}'} p'_u \cdot c'_u.\end{aligned}$$

Similarly, we can lower bound $\mathbb{E}[\text{OPT}']$ using Lemma 2.9:

$$\mathbb{E}[\text{OPT}'] \geq \sum_{v\in VC} \mathbb{E}[\text{OPT}'_{H'_v\cup\{v'\}}] + \sum_{u\in V'\setminus\mathcal{H}'} \mathbb{E}[\text{OPT}'_{\{u\}}] \geq \sum_{v\in VC} \mathbb{E}[\text{OPT}'_{H'_v\cup\{v'\}}] + \sum_{u\in V'\setminus\mathcal{H}'} p'_u \cdot c'_u. \tag{9}$$

Since the term $\sum_{u \in V' \setminus \mathcal{H}'} p'_u \cdot c'_u$ shows up in both inequalities, it remains, for each $v \in VC$, to bound $c'_{v'} + \sum_{u \in H'_v} p'_u \cdot c'_u$ in terms of $\mathbb{E}[\mathrm{OPT}'_{H'_v \cup \{v'\}}]$. By definition of H'_v, we have $(1 - p'_{v'}) \cdot c'_{v'} = \sum_{u \in H'_v} (1 - p'_u) \cdot c'_u$, which implies

$$c'_{v'} + \sum_{u \in H'_v} p'_u \cdot c'_u = p'_{v'} \cdot c'_{v'} + \sum_{u \in H'_v} c'_u. \tag{10}$$

The value $\mathbb{E}[\mathrm{OPT}'_{H'_v \cup \{v'\}}]$ corresponds to the expected optimum for the subproblem which considers the subgraph induced by $H'_v \cup \{v'\}$ (which is a star), uses p'_u as the mandatory probability for each $u \in H'_v \cup \{v'\}$, and uses c'_u as the query cost of each $u \in H'_v \cup \{v'\}$. For this subproblem, $c'_{v'} + \sum_{u \in H'_v} p'_u \cdot c'_u$ corresponds to the expected cost of the vertex cover-based algorithm that queries vertex cover $\{v'\}$ in the first stage. Furthermore, $p'_{v'} \cdot c'_{v'} + \sum_{u \in H'_v} c'_u$ corresponds to the expected cost of the vertex-cover based algorithm that queries vertex cover H'_v in the first stage. In summary, we have a star orientation subproblem for which both vertex cover-based algorithms (querying the leaves or the center first) have the same expected cost (cf. Equation (10)). As a major technical step, we show that BestVC is $\frac{4}{3}$-competitive for such subproblems, which implies

$$c'_{v'} + \sum_{u \in H'_v} p'_u \cdot c'_u \leq \frac{4}{3} \cdot \mathbb{E}[\mathrm{OPT}'_{H'_v \cup \{v'\}}].$$

A corresponding lemma is proven in the full version. We remark that the lemma requires $p'_{v'}$ to be independent of each p'_u with $u \in H'_v$; otherwise the subproblem does not correspond to the star orientation problem. As the input graph is bipartite, such independence follows by definition.

Using this inequality and Equation (9), we conclude that BestVC is 4/3-competitive.

$$\begin{aligned}\mathbb{E}[\textsc{BestVC}] &= \sum_{v \in VC} \left(c'_{v'} + \sum_{u \in H'_v} p'_u \cdot c'_u \right) + \sum_{u \in V' \setminus \mathcal{H}'} p'_u \cdot c'_u \\ &\leq \frac{4}{3} \cdot \sum_{v \in VC} \mathbb{E}[\mathrm{OPT}'_{H'_v \cup \{v'\}}] + \sum_{u \in V' \setminus \mathcal{H}'} p'_u \cdot c'_u \leq \frac{4}{3} \cdot \mathbb{E}[\mathrm{OPT}'] = \frac{4}{3} \cdot \mathbb{E}[\mathrm{OPT}] \quad \blacktriangleleft\end{aligned}$$

4.2 An Almost Best Possible Algorithm for Orienting a Hyperedge

▶ **Theorem 4.2.** BestVC *has a competitive ratio at most* $\min\{\frac{4}{3}, \frac{n+1}{n}\}$ *for the hypergraph orientation problem on a single hyperedge with* $n \geq 2$ *vertices and uniform query costs. For a hyperedge with only two vertices, the algorithm is* 1.207*-competitive.*

Our analysis improves upon a $(n+1)/n$-competitive algorithm by Chaplick et al. [16] in case that the hyperedge has two or three vertices. Moreover, we show that this is near-optimal: It is not hard to show a matching lower bound for two vertices and, due to Theorem 2.3 and the theorem below, this is the best possible for three vertices, and in general the difference between the upper and lower bounds is less than 4%.

▶ **Theorem 4.3.** *Any algorithm for orienting a single hyperedge with* $n + 1 \geq 2$ *vertices has competitive ratio at least* $n^2/(n^2 - n + 1)$*, even for uniform query costs.*

Note that Theorem 4.2 is in contrast to the problem of orienting a hyperedge with arbitrary query costs: In this setting, [16] showed that the algorithm is 1.5-competitive, which matches the corresponding lower bound for vertex cover-based algorithms of Theorem 2.6.

5 Conclusion

In this paper, we present algorithms for the (hyper)graph orientation problem under stochastic explorable uncertainty. It remains open to determine the competitive ratio of BestVC for the general (hyper)graph orientation problem, and to investigate how the algorithm behaves if it has to rely on an α-approximation to solve the vertex cover subproblem. In this context, one can consider the resulting algorithm as a standalone algorithm, or as a subroutine for Threshold. Our analysis suggests that, to achieve a competitive ratio better than 1.5, algorithms have to employ more adaptivity; exploiting this possibility remains an open problem. Finally, it would be interesting to characterize the vertex cover instances arising in our Threshold algorithm. In addition to the relevance from a combinatorial point of view, such a characterization may allow an improved α-approximation algorithm for those instances.

References

1 Marek Adamczyk, Maxim Sviridenko, and Justin Ward. Submodular stochastic probing on matroids. *Math. Oper. Res.*, 41(3):1022–1038, 2016.

2 Susanne Albers and Alexander Eckl. Explorable uncertainty in scheduling with non-uniform testing times. In *WAOA 2020: 18th International Workshop on Approximation and Online Algorithms*, 2021.

3 Luciana Arantes, Evripidis Bampis, Alexander V. Kononov, Manthos Letsios, Giorgio Lucarelli, and Pierre Sens. Scheduling under uncertainty: A query-based approach. In *IJCAI 2018: 27th International Joint Conference on Artificial Intelligence*, pages 4646–4652, 2018. doi:10.24963/ijcai.2018/646.

4 Sepehr Assadi, Deeparnab Chakrabarty, and Sanjeev Khanna. Graph connectivity and single element recovery via linear and OR queries. In *ESA 2021: 29th Annual European Symposium on Algorithms*, volume 204 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

5 Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem with (very) few queries. *ACM Trans. Economics and Comput.*, 7(3):16:1–16:19, 2019.

6 Evripidis Bampis, Christoph Dürr, Thomas Erlebach, Murilo S. de Lima, Nicole Megow, and Jens Schlöter. Orienting (hyper)graphs under explorable stochastic uncertainty. *CoRR*, abs/2107.00572, 2021. URL: https://arxiv.org/abs/2107.00572.

7 Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012.

8 Nikhil Bansal and Viswanath Nagarajan. On the adaptivity gap of stochastic orienteering. *Math. Program.*, 154(1-2):145–172, 2015.

9 Reuven Bar-Yehuda, Keren Bendel, Ari Freund, and Dror Rawitz. Local ratio: A unified framework for approxmation algrithms. In memoriam: Shimon Even 1935-2004. *ACM Comput. Surv.*, 36(4):422–463, 2004. doi:10.1145/1041680.1041683.

10 Reuven Bar-Yehuda and Shimon Even. On approximating a vertex cover for planar graphs. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *STOC 1982: 14th Annual ACM Symposium on Theory of Computing*, pages 303–309. ACM, 1982. doi:10.1145/800070.802205.

11 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. In Anna R. Karlin, editor, *ITCS 2018: 9th Innovations in Theoretical Computer Science Conference*, volume 94 of *LIPIcs*, pages 38:1–38:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.38.

12 Soheil Behnezhad, Alireza Farhadi, MohammadTaghi Hajiaghayi, and Nima Reyhani. Stochastic matching with few queries: New algorithms and tools. In *SODA 2019: 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2855–2874. SIAM, 2019.

13 Avrim Blum, John P. Dickerson, Nika Haghtalab, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance is almost bliss: Near-optimal stochastic matching with few queries. *Oper. Res.*, 68(1):16–34, 2020.

14 Richard Bruce, Michael Hoffmann, Danny Krizanc, and Rajeev Raman. Efficient update strategies for geometric computing with uncertainty. *Theory of Computing Systems*, 38(4):411–423, 2005. doi:10.1007/s00224-004-1180-4.

15 Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

16 Steven Chaplick, Magnuús M. Halldórsson, Murilo S. de Lima, and Tigran Tonoyan. Query minimization under stochastic uncertainty. In Y. Kohayakawa and F. K. Miyazawa, editors, *LATIN 2020: 14th Latin American Theoretical Informatics Symposium*, volume 12118 of *Lecture Notes in Computer Science*, pages 181–193. Springer Berlin Heidelberg, 2020.

17 Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *ICALP 2009: 36th International Colloquium on Automata, Languages and Programming, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 266–278. Springer, 2009. doi:10.1007/978-3-642-02927-1_23.

18 Xi Chen, Amit Levi, and Erik Waingarten. Nearly optimal edge estimation with independent set queries. In Shuchi Chawla, editor, *SODA 2020: Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms*, pages 2916–2935. SIAM, 2020. doi:10.1137/1.9781611975994.177.

19 Miroslav Chlebik and Janka Chlebíková. The complexity of combinatorial optimization problems on d-dimensional boxes. *SIAM Journal on Discrete Mathematics*, 21(1):158–169, 2007.

20 Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.*, 33(4):945–964, 2008.

21 Christoph Dürr, Thomas Erlebach, Nicole Megow, and Julie Meißner. An adversarial model for scheduling with testing. *Algorithmica*, 82(12):3630–3675, 2020.

22 Thomas Erlebach and Michael Hoffmann. Minimum spanning tree verification under uncertainty. In D. Kratsch and I. Todinca, editors, *WG 2014: International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 8747 of *Lecture Notes in Computer Science*, pages 164–175. Springer Berlin Heidelberg, 2014. doi:10.1007/978-3-319-12340-0_14.

23 Thomas Erlebach, Michael Hoffmann, and Murilo S. de Lima. Round-competitive algorithms for uncertainty problems with parallel queries. In Markus Bläser and Benjamin Monmege, editors, *STACS 2021: 38th International Symposium on Theoretical Aspects of Computer Science*, volume 187 of *LIPIcs*, pages 27:1–27:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.27.

24 Thomas Erlebach, Michael Hoffmann, Murilo S. de Lima, Nicole Megow, and Jens Schlöter. Untrusted predictions improve trustable query policies. *CoRR*, abs/2011.07385, 2020. URL: https://arxiv.org/abs/2011.07385.

25 Thomas Erlebach, Michael Hoffmann, Danny Krizanc, Matús Mihalák, and Rajeev Raman. Computing minimum spanning trees with uncertainty. In *STACS'08: 25th International Symposium on Theoretical Aspects of Computer Science*, volume 1 of *LIPIcs*, pages 277–288. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2008. doi:10.4230/LIPIcs.STACS.2008.1358.

26 Tomás Feder, Rajeev Motwani, Liadan O'Callaghan, Chris Olston, and Rina Panigrahy. Computing shortest paths with uncertainty. *Journal of Algorithms*, 62(1):1–18, 2007. doi:10.1016/j.jalgor.2004.07.005.

27 Tomás Feder, Rajeev Motwani, Rina Panigrahy, Chris Olston, and Jennifer Widom. Computing the median with uncertainty. *SIAM Journal on Computing*, 32(2):538–547, 2003. doi:10.1137/S0097539701395668.

28 Jacob Focke, Nicole Megow, and Julie Meißner. Minimum spanning tree under explorable uncertainty in theory and experiments. *ACM J. Exp. Algorithmics*, 25:1–20, 2020. doi:10.1145/3422371.

29 András Frank. *Connections in Combinatorial Optimization*. Oxford University Press, USA, 2011.

30 John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed Bandit Allocation Indices*. Wiley, 2nd edition, 2011.

31 Marc Goerigk, Manoj Gupta, Jonas Ide, Anita Schöbel, and Sandeep Sen. The robust knapsack problem with queries. *Computers & Operations Research*, 55:12–22, 2015. doi:10.1016/j.cor.2014.09.010.

32 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.

33 Anupam Gupta, Haotian Jiang, Ziv Scully, and Sahil Singla. The Markovian price of information. In *IPCO 2019: 20th International Conference on Integer Programming and Combinatorial Optimization*, volume 11480 of *Lecture Notes in Computer Science*, pages 233–246. Springer, 2019.

34 Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Running errands in time: Approximation algorithms for stochastic orienteering. *Math. Oper. Res.*, 40(1):56–79, 2015.

35 Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *IPCO 2013: 16th International Conference on Integer Programming and Combinatorial Optimization*, volume 7801 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 2013.

36 Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Algorithms and adaptivity gaps for stochastic probing. In *SODA 2016: 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1731–1747. SIAM, 2016.

37 Manoj Gupta, Yogish Sabharwal, and Sandeep Sen. The update complexity of selection and related problems. *Theory of Computing Systems*, 59(1):112–132, 2016. doi:10.1007/s00224-015-9664-y.

38 Magnús M. Halldórsson and Murilo Santos de Lima. Query-competitive sorting with uncertainty. *Theor. Comput. Sci.*, 867:50–67, 2021. doi:10.1016/j.tcs.2021.03.021.

39 Eran Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computing*, 31(5):1608–1623, 2002.

40 Simon Kahan. A model for data in motion. In *STOC'91: 23rd Annual ACM Symposium on Theory of Computing*, pages 265–277, 1991. doi:10.1145/103418.103449.

41 Sanjeev Khanna and Wang-Chiew Tan. On computing functions with uncertainty. In *PODS'01: 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 171–182, 2001. doi:10.1145/375551.375577.

42 Retsef Levi, Thomas L. Magnanti, and Yaron Shaposhnik. Scheduling with testing. *Manag. Sci.*, 65(2):776–793, 2019.

43 Will Ma. Improvements and generalizations of stochastic knapsack and Markovian bandits approximation algorithms. *Math. Oper. Res.*, 43(3):789–812, 2018.

44 Hanna Mazzawi. Optimally reconstructing weighted graphs using queries. In Moses Charikar, editor, *SODA 2010: 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 608–615. SIAM, 2010. doi:10.1137/1.9781611973075.51.

45 Nicole Megow, Julie Meißner, and Martin Skutella. Randomization helps computing a minimum spanning tree under uncertainty. *SIAM Journal on Computing*, 46(4):1217–1240, 2017. doi:10.1137/16M1088375.

46 Arturo I. Merino and José A. Soto. The minimum cost query problem on matroids with uncertainty areas. In *ICALP 2019: 46th International Colloquium on Automata, Languages, and Programming*, volume 132 of *LIPIcs*, pages 83:1–83:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

47 George L. Nemhauser and Leslie E. Trotter Jr. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.

48 Noam Nisan. The demand query model for bipartite matching. In *SODA 2021: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms*, pages 592–599. SIAM, 2021. doi:10.1137/1.9781611976465.36.

49 Chris Olston and Jennifer Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *VLDB 2000: 26th International Conference on Very Large Data Bases*, pages 144–155, 2000. URL: http://ilpubs.stanford.edu:8090/437/.

50 Herbert E. Robbins. A theorem on graphs, with an application to a problem of traffic control. *The American Mathematical Monthly*, 46(5):281–283, 1939. URL: http://www.jstor.org/stable/2303897.

51 Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing exact minimum cuts without knowing the graph. In Anna R. Karlin, editor, *ITCS 2018: 9th Innovations in Theoretical Computer Science Conference*, volume 94 of *LIPIcs*, pages 39:1–39:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.39.

52 Alexander Schrijver. *Combinatorial Optimization − Polyhedra and Efficiency*. Springer, 2003.

53 Sahil Singla. The price of information in combinatorial optimization. In *SODA 2018: 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2523–2532. SIAM, 2018. doi:10.1137/1.9781611975031.161.

54 William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

55 Martin Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–54, 1979.

56 Mihalis Yannakakis and Fanica Gavril. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 38(3):364–372, June 1980. doi:10.1137/0138030.